Title: Small burst data (SBD) satellite communications

Author(s): Saari, Alexandra
Frigo, Janette Rose
Mccabe, Kevin Peter
Gard, Marvin Orrmond
Sevanto, Sanna Annika
Warniment, Adam
Borges, Louis
Proicou, Michael Chris

Intended for: SULI Deliverable
Report

Issued: 2014-08-14

LOS ALAMOS NATIONAL LABORATORY

# Small burst data (SBD) satellite communications

**Alexandra Saari, Mike Proicou, Janette Frigo, Kevin McCabe, Marvin Gard, Sanna Sevanto, Adam Warniment, Louis Borges**

**8/13/2014**

Using developmental Sensor System hardware, we tested and deployed prototypes that transmitted data from remote sites to the user's computer through the Iridium satellite network. Once the existing software and hardware proved functional and reliable, we were to perform the software development to implement 2-way communication between the remote unit and the user's computer via the 2-way satellite modem link. Upon successful completion of this task, continue to develop additional functionality such as State-of-Health (SOH) monitoring. SOH monitoring in this case consisting of detecting the on-board battery voltage and the processor temperature every 24 hours, checking their values and sending a message to the host indicating whether the system has an acceptable SOH. Upon request, SOH values for battery voltage and processor temperature would be polled and sent to the host.

# Small burst data (SBD) satellite communications

Alexandra Saari, Mike Proicou, Janette Frigo, Kevin McCabe, Marvin Gard, Sanna Sevanto, Adam Warniment, Louis Borges

Critical hardware and scientific systems located in remote regions always require discussion of how best the operator can control and monitor the equipment. Generally, either someone must be on site, or the research team must utilize some remote monitoring scheme utilizing radio, cellular or satellite telemetry. Due to large-scale industrial and military interests such as oil and mining operations, or war, that often take place outside of established or intact infrastructure, there has been significant development of commercial systems that can accommodate these customers. However, these systems are often priced with a wealthy client base using tightly controlled acquisition methods in mind. In other words, the systems are expensive because the entities buying them have lots of money, and there is little competition. For concerns with smaller budgets that could use these systems, like science or small business, the cost of entry is prohibitive. This project seeks to address these smaller scale users by creating a low cost alternative that is versatile and configurable by the user for unconventional or innovative purposes.

This is an ongoing project, led by Janette Frigo. My tasks, as discussed through this document, were to work with Mike Proicou, the lead software developer, to test the existing hardware and software for functionality and implement two way communications between the hardware and the Iridium network.

## HARDWARE INFORMATION

The Iridium satellite constellation is a global array of 66 Low Earth Orbit (LEO) satellites that provide voice and data communication to any point on earth, provided the ground antenna has an unobstructed view of the sky, and the US does not have a trade embargo in the callers region. The system became operational in 1998 and boasted that “Iridium’s satellites are the only ones in the cosmos that communicate with one another as well as bounce signals back to Earth.”[1] As a consequence, an Iridium handset anywhere in the world can communicate with another Iridium handset anywhere in the world.

The board is an embedded system using a Los Alamos (LANL) designed circuit board equipped with a variety of interfaces including RS232 Serial and USB, onboard memory, and an SD card slot. Due to the remote nature of the deployed system, power consumption must be minimized while retaining a high level of functionality. To address this, the board is administered through a Real Time Operating System (RTOS) running on a ST Microelectronics ARM M3 cortex processor (STM32F407ZGT6)[2] that draws as little as 238 µA. The processor is

[1] (Mellow, 2004)
[2] (STMicroelectronics, 2014)

programmed in the C language using Keil's uVision 4 IDE[3] and Keil's uLink-ME programmer. Planned features include radio and direct sensor inputs. The RTOS is the RTX system from Keil[4]. Satellite communications were accomplished through an Iridium Short Burst Data (SBD) modem from Sutron.

The SBD modem communicates with the satellite network and the mobile hardware. It sends to, and receives data from the network directly in the form of SBD packets. The modem can receive up to 270 bytes and transmit up to 340 bytes to the network per communication. Data to or from the satellite network are received and processed by an Iridium "gateway" location, the data are then processed by a reseller that distributes the information to the user through some type of prearranged network interface (email attachment, SMS text message, website, etc…). To send data to their modem, the users must send their data to the reseller via a prearranged method, the reseller passes the data to Iridium, Iridium communicates with the satellite network, and the modem receives the users' data.

Iridium produces the modems that communicate with their network and distributes them to their resellers as sealed "black boxes" which are then incorporated into the reseller's hardware or resold directly to the end user. The reseller often offers some combination of gateway access and hardware as a bundle to the end user, though some resellers will sell gateway access to any existing Iridium modem.

## TESTING: PART 1, "ENVIRONMENTAL DATA MONITORING BOARD"

An earlier iteration of the project used an ARM 7 development board paired with a LANL designed board providing sensor, power and serial interfaces between the development board and modem. The RTX system administered the system using software developed by Mike Proicou and provided data transmission capability. This board, known as the Environmental Data Monitoring Board was designed to monitor the depth of a remotely located water tank, and transmit the results at a predetermined interval to the Iridium network. Initially, we needed to test this board and its software for functionality.

There were no locations in the immediate work areas to position the Iridium antenna for reliable reception. So to start, I connected to the board's serial port with a laptop to monitor the board's messages to the modem. The "sensors" in this system consisted of conductors mounted at different heights in a water tank and attached to the sensor inputs on the board. The water tank would be grounded to the board. When the sensors were polled, the sensor leads would be floated to a 12 volt "high" state. If a lead were submerged, the water would ground the sensor lead to a "low" state and it would be recorded by the microprocessor. The number of sensors that registered "low" during polling would indicate the level of water in the tank. At Mike's suggestion, I set up a cup of water with wire leads at different heights and connected to the sensor inputs. Initial testing showed that the code attempted to poll the sensors and initiated a modem transfer of the data as expected. Numerous attempts to trigger the water level sensor inputs produced no response in the modem's transmitted data summary. Careful examination of the code indicated no obvious data processing issues. However, after conferring with Adam Warniment, and examination of the hardware, it was discovered that an optocoupler had been

[3] (Keil Software and Design, 2009)
[4] (Keil Tools by ARM)

installed incorrectly (reverse orientation), and was preventing the proper powering of the sensor array.

Louis Borges remounted the optocoupler as described in the original prototype design, but the part proved to have been irreparably damaged by its reverse orientation. After further discussion with Adam, for the purposes of testing the software and functioning hardware, the damaged optocoupler was bypassed. After retesting the system with these adjustments, the system transmitted data as expected, showing each sensor being activated.

After replacing the optocoupler , the system was retested, but again did not indicate any sensor inputs. Examination of the circuit with an oscilloscope showed the optocoupler receiving a trigger impulse of approximately 50 microseconds with no response. After examination of the data sheet for the replaced optocoupler (model G3VM-61GR1), it was found that the component had a minimum turn-on time of 1.5 milliseconds. The control software was updated with a 5 millisecond delay after triggering the sensors to allow the optocoupler time to respond, and the system began reporting sensor inputs correctly, although in what appeared to be a reverse configuration. The system appeared to function correctly with the water floated to 12 volts during polling, bringing the submerged sensors high instead of low.

At the conclusion of lab testing, the system was given to Marvin Gard for a test installation on a small water tank located on a small ranch outside of Santa Fe. Upon installation, the system began broadcasting data sets as expected. However, the data was inconsistent with the actual observed water level. Kevin and I travelled to Santa Fe to inspect the system with Marvin and found the sensor assembly was accumulating excess moisture and the sensors appeared to be energizing each other through the water on the assembly. Kevin examined the schematic of the prototype board and found that the system, while functional in either configuration, should be connected with the water tank grounded instead of floated with 12 volts. Time of day and electric storms prevented further work on the system that evening, so I returned the following morning with Janette. After reversing the voltage configuration, the sensors indicated they were all submerged, despite the tank being almost empty. Taking the lesson from the earlier optocoupler into account, I thought the components between the power supply and sensor inputs may also need additional rise time. So may not be energized long enough to bring the sensors high. I changed the delay after polling from 5 ms to 10 ms and the system appeared to be working correctly. Excess moisture continues to be an issue, so Marvin is examining the sensor assembly for a future redesign.

## TESTING: PART 2, "LOGGER BOARD"

The "Logger board" is an early version of the envisioned system containing only the ARM M3 Cortex processor, power management hardware, programming header, RS232 Serial port, and a simple 3 wire serial header for use with a Campbell Scientific CR1000 data logger[5]. Software for the logger is written in CRBasic. CRBasic can be written and error checked with Campbell Scientific's IDE or just written in Notepad on Windows machines. The source code is uploaded to the logger through Campbell Scientific's communication software (in this case, PC200W) and a standard serial cable. Mike also wrote the software for both the prototype board and logger.

[5] (Campbell Scientific, 2013)

This version of the system had already been deployed to two field locations on environmental monitoring stations in Abisko Sweden, and in Alaska at (roughly) 66.508231 latitude, and -157.497658 longitude. The Abisko station had been damaged by snow while servicing, and needed data transmission adjustments as the data was overflowing the message limit for SBD transmissions. The Alaska station stopped transmitting after it was completely destroyed by wildlife.

A new logger board was prepared for Abisko and conformal coated for weatherization. Mike adjusted the Campbell code to send an "A" and "B" message to control message lengths. I tested the new board and code to ensure everything worked as expected. There was a small error in the code that kept the system from powering up properly, but after that was corrected everything seemed okay. We gave the board to Sanna Sevanto who delivered it to the station in Sweden.

The station in Sweden had the same CR1000 data logger as we had for testing, but also had a pair of 16/32 channel multiplexers attached to the logger. Everything was written to expect 64 channels of data output for transmission, but the CR1000 and multiplexers had also received water damage during servicing. The CR1000 was swapped for a new unit, and the code was rewritten on site to bypass polling of the damaged channels on the multiplexers, giving 53 channels of output for transmission. Unfortunately, the system began reporting inconsistent temperature values upon activation. Mike suspected the problem was a result of not adjusting the size of the data array that stored the sensor outputs after reducing the number of channels. He suggested each set of data was being written to the array at an offset due to the initial buffers not being completely full. After examining the rewritten code and testing it both with fixes and without, I found the original code to be robust and tolerant of length errors. After numerous attempts at corrupting the incoming data, interfering with Iridium communication, and fluctuating power supply, I was unable to break it and the system continued to perform as designed. We contacted Sanna and asked her to check the wiring in the system and sensor attachments. Afterwards, the system began broadcasting again with appropriate data transmissions.

## IMPLEMENTING TWO-WAY COMMUNICATION

Communication with Iridium SBD modems is accomplished through the internet as described previously. In the case of the Sutron modem, modems can be contacted by sending an email to data@sbd.iridium.com with the IMEI # of the modem as the subject line, and an attachment containing the data wishing to be sent. The attachment should be a text file saved with a "sbd" extension, for example "dataformodem.sbd". This is the same procedure outlined in the Iridium documentation, so is not unique to the Sutron product. Iridium SBD modems use an extended set of standard modem AT commands for control. For example, the ring alert for a standard modem is simply, "+RING", but for the SBD modem, this is changed to "+SBDRING". These commands and their functions are described in detail in the modem vendor and Iridium product documentations.

The RTOS is a multithreaded system that uses a “round robin” technique to circulate through assigned tasks, allowing individual tasks to take different amount of time or resources without delaying (within a designed tolerance) the execution of the other running tasks. This system works well with the Interface Board as it needs to coordinate the communication between multiple pieces of controlled equipment and the attached communication devices so that all data and commands are delivered as expected. The technique applied in this case is visualized in figure 1.

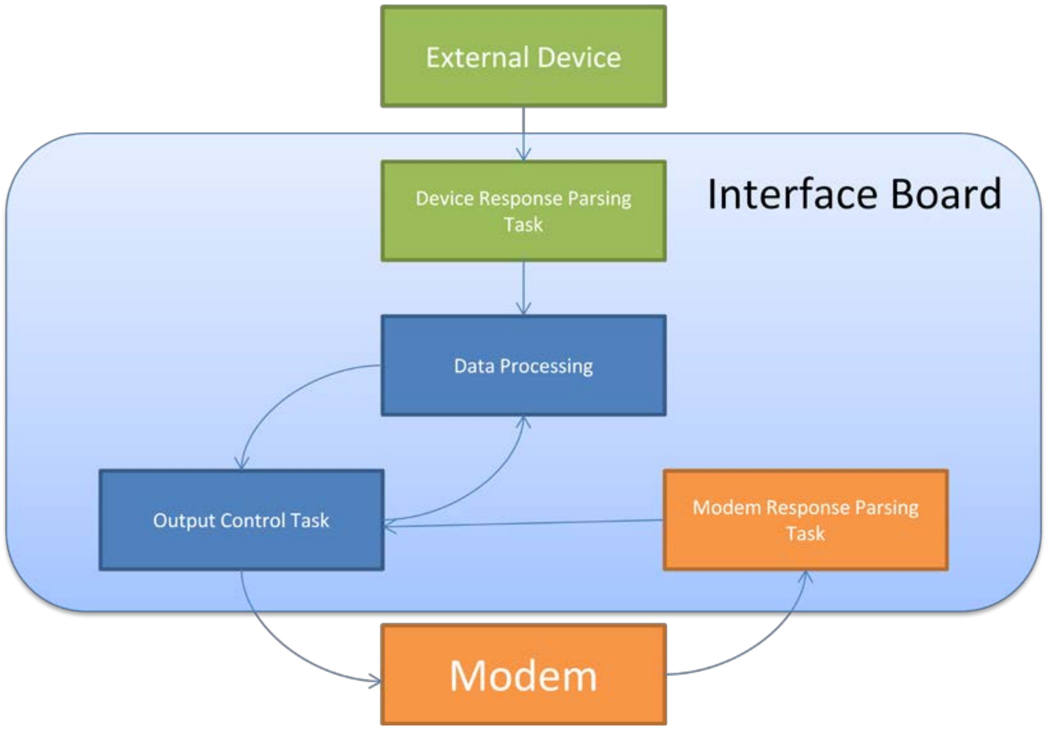

Figure 1

The external device in this case is a Campbell Scientific CR1000 data logger and only provides output to the Interface Board. The board has a task that exclusively listens for data from the logger. This task sends any data it receives to the RTOS for processing and then requests an appropriate transmission from the Output Control Task (OCT). From the modem, another task, exclusively listens for modem responses and sends that data to the Output Control Task. The OCT collects responses from the modem and responds or, if necessary, sends the received data to be processed. The OCT’s main purpose is to collect any requests for transmission and ensure that they are sent to the modem one at a time. Each attached device should only have one task that responds to it for input or output. This prevents devices from mixing data streams and receiving or sending garbled data.

Implementing this method so that it is responsive and robust enough to handle variable network conditions is not trivial and is an ongoing project beyond the time frame specified in this internship. What has been accomplished as of this writing  is the determination of the necessary AT codes for receiving modem terminated messages (MTM), and retrieving them from the modem data buffers. Further, the existing command parser written by Mike Proicou has been re-factored to recognize and respond to these new commands. The development of the output task to control the requests from the various devices and tasks is ongoing, and still not functional to an acceptable level of reliability. The documentation available for developing a field application such as this is not published and is proprietary to Iridium Satellite LLC and the vendor (in this case, the Sutron company). Consequently, the command documentation has been proven to be not entirely accurate as to the expected responses from the modem.

To handle the lack of documentation, experiments were designed to fill in knowledge gaps. Using a serial interface with the modem, we were able to observe the modem responses directly, as it was presented with data from the Iridium network and the terminal. Unfortunately, the available locations to perform these experiments did not have ideal signal strength from the network, which cost time. However, it did allow us to gain experience with poor network conditions, which should be helpful with the development of the Interface Board's software.

Future work will continue the development and testing of the board's software implementing reliable two-way communication with the hardware. Later prototypes of the Interface Board will have additional I/O support including onboard radio communications. Once the basic two-way communication has been enabled, we will implement the onboard radio system's mesh networking capability. Further features and functionality for the Interface Board will be added as requested.

## ACKNOWLEDGEMENTS

I would like to thank my LANL mentor, Joysree Aubrey; my project mentors Janette "Jan" Frigo, and Kevin McCabe for designing the Interface Board and heading this project; Mike Proicou for writing the main code and helping me learn about satellites, RTOS's and embedded systems; Adam Warniment and Louis Borges for help with hardware troubleshooting and repair; and finally, Marvin Gard and Sanna Sevanto for field operations and testing. They have all been exceedingly helpful and welcoming this summer in making The Lab an incredible place to work.

## WORKS CITED

Campbell Scientific. (2013, May). CR1000 Measurement and Control System Revision: 5/13. Campbell Scientific, Inc.

Keil Software and Design. (2009). *Keil Tools by ARM, Getting Started, Creating Applications with µVision®4.* Keil, Tools by ARM, and ARM Ltd.

Keil Tools by ARM. (n.d.). *RL-ARM User's Guide.* Retrieved August 12, 2014, from Keil Tools by ARM: http://www.keil.com/support/man/docs/rlarm/rlarm_ar_artxarm.htm

Mellow, C. (2004, September). *The Rise and Fall and Rise of Iridium*. Retrieved July 30, 2014, from Air & Space Magazine: http://www.airspacemag.com/space/the-rise-and-fall-and-rise-of-iridium-5615034/?all

STMicroelectronics. (2014, May). RM0090 Reference Manual. *STM32F405xx/07xx, STM32F415xx/17xx, STM32F42xxx and STM32F43xxx advanced ARM®-based 32-bit MCUs* .